Remarks

Status of application

Claims 1-3, 5-8, 10-19, 21-24, and 26-30 were examined and stand rejected in view of prior art, as well as stand rejected for technical issues. The claims have been amended to further clarify Applicant's invention. In view of the amendments made and the following remarks, reexamination and reconsideration are respectfully requested.

General

A. Specification

The Examiner has objected to the Specification as not including appropriate references to several trademarks. Applicant notes that in a prior Office Action dated February 8, 2007, the Examiner made the same objection to the Specification as not including references to several trademarks, including Btrieve, Java, and Unix. In response, Applicant amended the specification to include references to these trademarks as requested by the Examiner. These amendments to the Specification were initially rejected by the Examiner as not being properly marked; however the resubmitted (marked) Amendments to the Specification with markings were filed by Applicant in an Amendment After Final filed on May 8, 2007. In the Advisory Action dated July 19, 2007 the Examiner agreed to enter the Amendments to the Specification. Accordingly, Applicant respectfully believes the objection to the Specification is overcome by Applicant's previously filed Amendments to the Specification.

B. Drawings

The Examiner has objected to the drawings under 37 CFR 1.83(a) stating that the drawings must show every feature of the invention specified in claims 1-3, 5-8, and 10-16. The Examiner has not, however, indicated any specific features of these claims which the Examiner does not believe are found in the drawings, thus making it unclear what specific features of these claims the Examiner believes are not found in the drawings.

Applicant respectfully believes the claimed features are illustrated in the

drawings. For example, as to Applicant's amended claim 1 the claimed features are shown in the drawings as follows: In a database system employing a transaction log (Fig. 3 at 340 (database system) and at 350 (log manager)), an improved method for restoring a database to a consistent version supporting read-only uses (Fig. 4A at 401-405, Fig. 4B at 411-413), the method comprising: providing a shared cache storing database blocks in memory of the database system (Fig. 1 at 102 (RAM), 103 (ROM); Fig. 3 at 340 (database system) and at 360 (cache manager)), in response to a read-only transaction of a given database, creating a cache view of the given database using the given database's transaction log by logically undoing transactions which have begun but have yet to commit, said cache view comprising particular database blocks in the shared cache that record a transactionally consistent version of the given database at a given point in time supporting read-only uses (Fig. 3 at 340, 350 (log manager), 360 (cache manager); Fig. 4A at 402, 403, 404, 405; Fig, 4B at 411, 412, 413), temporarily storing any database blocks that overflow said shared cache in a temporary database (shadow cache) during use of the cache view by the read-only transaction (Fig. 3 at 340, 360 (cache manager); Fig. 4A at 401); and performing the read-only transaction using the cache view and returning results of the read-only transaction (Fig. 3 at 310 (clients), 320, 340, 360 (cache manager), 389 (cursor execution module); Fig. 4A at 405).

C. Claim Objections.

The Examiner has objected to the use of the term "preserving" in Claim 1 on the basis that it is unclear how the logical undo operation is preserved. Applicant respectfully believes that the description of how the cache view is created and maintained in cache during the pendency of the read-only transaction is clear; however Applicant has amended Claim 1 to remove the term "preserving" and to more clearly indicate that the "cache view" is created by logically undoing incomplete transactions and maintaining this cache view in memory (i.e., the shared cache and/or temporary database) for use by the read only transaction.

The Examiner has also objected to claims 1 and 17 as comprising a recitation of use without setting for the steps involved in the process. Applicant has amended claims 1 and 17 in an effort to more clearly articulate that in Applicant's claimed invention creates

and maintains in memory a "cache view" which comprises a transactionally consistent version of a given database at a particular point in time. The cache view is created by logically undoing transactions which have begun but have yet to commit and is maintained in cache for use by read-only transactions.

Based on the foregoing amendments, Applicant respectfully believes the Examiner's objections to claims 1 and 17 are overcome.

D. Section 101 rejection

Claims 1-3, 5-8, 10-19, 21-24, and 26-30 stand rejected under 35 U.S.C. 101 on the basis of non-statutory subject matter. Here, the Examiner states that Applicant's claimed invention does not produce a tangible result because the claim limitations remain in the abstract and, thus, fails to achieve the required status of having real world value. Applicant's claimed invention provides for maintaining a "cache view" (which represents a transactionally-consistent prior state of the database) in cache and using this view for performing read-only transactions (e.g., DSS, reporting, data mining, or the like) without blocking the performance of other transactions. Applicant has amended its independent claims to including limitations of performing the read-only transaction using the cache view and returning results; thereby describing a concrete and tangible result.

The Examiner additionally rejects claims 17-19, 21-24, and 26-30 on the basis that claim 17 recites a computer program *per se* and hence is non-statutory. Although Applicant respectfully believes that the Examiner has incorrectly construed Applicant's specification as stating that the elements of Applicant's invention can only be implemented in software, Applicant has amended claim 17 to include limitations of a computer having a processor and memory. These limitations are supported in Applicant's specification. For example, paragraph [0033] of Applicant's specification states as follows: "...the corresponding apparatus element may be <u>configured in hardware, software, firmware or combinations thereof</u>" (Applicant's specification, paragraph [0033], emphasis added). Applicant's specification also describes in detail a computer hardware and software environment in which Applicant's invention may be implemented (Applicant's specification, paragraphs [0035]-[0045]; also see generally, Figs. 1 and 2). Moreover, Applicant states that the software (computer-executable instructions) direct

operation of a device under processor control, such as a computer (Applicant's specification, paragraph [0063]). As Applicant's claims 17-19, 21-24, and 26-30 define a useful machine or item of manufacture in terms of a hardware or hardware and software combination, Applicant respectfully believes that it defines a statutory product

Accordingly, it is respectfully submitted that in view of the above-mentioned amendments, the rejection of Applicant's claims 1-3, 5-8, 10-19, 21-24, and 26-30 under Section 101 on the basis of non-statutory subject matter is overcome.

Prior Art Rejections

A. First Rejection under 35 U.S.C. 103(a): Hayashi, Loaiza and Luo

Claims 1-3, 5-8, 10, 12, 13, 15-19, 21-24, 26, 28, and 29 stand rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent 5,715,447 of Hayashi et al., ("Hayashi") in view of U.S. Patent 6,618,822 to Loaiza et al., ("Loaiza") and further in view of U.S. Patent 6,990,503 to Luo et al. ("Luo"). The Examiner's rejection of Applicant's claim 1 as follows is representative:

> 1. In a database system employing a transaction log, an improved method for restoring databases to a consistent version, the method comprising:
>
> providing a shared cache storing database blocks for use by multiple databases (*See e.g. Hayashi Fig. 4 and col. 1, lines 30-41, "When a transaction accesses the database for some data through the database management system, the data may already be in a buffer shared by transactions due to another transaction that previously accessed the same data, or the data must be transferred from the database to the shared buffer"*);
>
> for a read-only transaction of a given database, creating a cache view of a given database using the given database's transaction log (*Hayashi does not teach creating a cache view using the given database's transaction log. However, Loaiza does, see col. 3, lines 12-28, 'The database view of a recovery log ('log view') essentially provides a virtual database table that is constructed using data retrieved from one or more recovery logs... A SQL statement can be written to access or manipulate data in the virtual rows and columns of the log view" where the claimed "cache view" is the referenced "log view, "the claimed "transaction log" is the referenced "recovery log," and the claimed "read-only transaction" is the referenced "SQL statement." Thus, it would have been obvious to one q/ ordinary skill in the database art at the time of the invention to combine the teachings of the cited references because Loaiza's teachings would have allowed Hayashi's method and system to gain a virtual view of the database regardless of system crashes or other transactions, see Loaiza col. 3, lines 12-28*), said cache

view comprising particular database blocks of the shared cache that record a view of a particular version of the database at a given point in time (*See e.g., Loaiza Table 1, col. 60, "Object ID Timestamp Block Addr." and col. 3, lines 52-56, "To provide relational access to the records in this recovery log, a log view is defined having virtual columns for each item of information sought for each log record" which shows that the "log view" (i.e. the "cache view') comprises database blocks, via the "Block Addr." that, using the "Timestamp," record a view of the database at a particular point in time. Loaiza does not teach using database blocks from a shared cache. However, Hayashi does, see Fig. 4, "shared buffer 17," referenced above. Thus, it would have been obvious to one of ordinary skill in the database art at the time of the invention to combine the teachings of the cited references because Hayashi's teachings would have allowed Loaiza's method and system to gain a common method of reusing database blocks, see Hayashi col. 1, lines 30-41*);

creating a shadow cache for storing any database blocks that overflow said cache view during use of the cache view by the read-only transaction (*See e.g. Hayashi Fig. 6 where, see col. 4, lines 8-18, "A buffer shared by the transactions is a bit map 30. The database 20 includes overflow pages 31. A database 20' is used to nonvolatilize the contents of the shared buffer. The bit map 30 controls overflow pages 31" where the claimed "shadow cache" is the referenced "overflow pages 31"*);

in conjunction with the cache view and the shadow cache, preserving a logical undo operation for the read-only transaction of the given database for logically undoing transactions which have begun but have yet to commit (*Hayashi does not teach preserving a logical undo operation. However, Luo does, see Fig. 5A and cal. 13, lines 16-21, "a logical undo is performed." Thus, it would have been obvious to one of ordinary skill in the database art at the time of the invention to combine the teachings of the cited references because Luo's teachings would have allowed Hayashi's method and system to gain the ability to rollback the user's view of the database without effecting the database, see Luo col. 13, lines 16-36*); and

performing the logical undo operation in order to reconstruct a transactionally consistent prior version of the given database upon starting the read-only transaction, thereby returning a result comprising a transactionally consistent version of the given database supporting read-only uses (*Hayashi does not teach performing a logical undo operation. However, Luo does, see Fig. 5A and col. 13, lines 16-21, "In the logical undo, the database system looks for another tuple in the join view JV that has the attribute values (1, 2). That other tuple is changed to the value (1, 1) for a logical undo of transaction T1, shown as 156 in FIG. 5A." Thus, it would have been obvious to one of ordinary skill in the database art at the time of the invention to combine the teachings of the cited references because Luo's teachings would have allowed Hayashi's method and system to gain the ability to rollback the user's view of the database without effecting the database, see Luo col. 13, lines 16-36*).

Applicant's invention provides for creating a "cache view" of a given database in

memory that can be used for performing read-only transactions in a manner that avoids the read-only transactions from being blocked by exclusive transactional locks of write transactions. Applicant's methodology also avoids having long duration read-only transactions block write transactions with long duration shared transactional locks. An important aspect of Applicant's invention is the use of logical undo operations for constructing the cache view, which is read-only view of a given database representing a transactionally consistent version of the given database at a particular point in time. In a given database environment, one may have multiple database users with intermingled write operations occurring against the database, some of which have committed while others are still pending. Applicant's approach provides a mechanism to reconstruct a **transactionally consistent "cache view"** of the database by logically undoing all incomplete transactions at the point in time when the cache view (i.e., a read-only view or version) is to be reconstructed (Applicant's specification, paragraphs [0082]-[0083]). Applicant's transactionally consistent "cache view" of the database may be used read-only transactions without blocking the write operations which may be concurrently occurring against the database.

The Examiner continues to rely on Hayashi, equating Applicant's claimed invention which provides the above-described "cache view" for performing read-only transactions and other decision support system (DSS) uses to Hayashi's solution which is focused on accelerating throughput for online transaction processing (OLTP) uses. To briefly recap, Hayashi is concerned with reducing the lock period of a shared buffer. Applicant's invention is not concerned with reducing lock periods. Rather the goal of the "cache view" created by Applicant's solution is to maintain a transactionally consistent view of a given database supporting read only uses. In contrast, Hayashi describes a database cache that shares cache blocks across multiple read/write transactions. Significantly, Hayashi's solution allows only one read-write view to be maintained in cache for a given database at one time, while Applicant's invention allows one or more read-only "cache views" of a given database at any point in time (given that such views are for read only purposes). Additionally, with Hayashi's solution, transactional consistency is maintained through the use of a traditional database lock manager which controls concurrent access to Hayashi's read-write cache. However, since Applicant's

cache views are read-only views (i.e., used by read-only transactions), a lock manager is not needed for concurrency control. For these and other reasons discussed in detail in Applicant's previously filed Appeal Brief (which is hereby incorporated by reference) Applicant respectfully believes Applicant's invention is distinguishable from Hayashi in a number of respects.

In an effort to cure some of the acknowledged deficiencies of Hayashi as to Applicant's claimed invention, the Examiner now adds the Loaiza and Luo references. However these references, even when combined, do not cure the deficiencies of Hayashi with respect to Applicant's invention as discussed in detail below.

The Examiner acknowledges that Hayashi does not teach creating a cache view using the database's transaction log and therefore references Loaiza for these teachings. However, Loaiza's solution is concerned with executing SQL queries against transactional log files. It does so by presenting the log files as tables as described, for example, as follows:

> The database view of a recovery log ("log view") essentially provides a virtual database table that is constructed using data retrieved from one or more recovery logs….The log view can be tailored to contain only columns of interest to the anticipated user. Thus, not all items of data from the recovery log necessarily needs to be included as a column in the log view. On the other hand, the log view may include columns that do not directly correspond to individual data items in the underlying recovery logs. This is useful, for example, if two or more data items from a recovery log are to be combined to form the contents of a log view column.
>
> (Loaiza, col. 3, lines 12-28)

Loaiza then goes on to describe the "log view" that is created from records in the recovery log as follows:

> To provide relational access to the records in this recovery log, a log view is defined having virtual columns for each item of information sought for each log record. The following is an example of a log view that may be defined for a recovery log:
>
> | Object ID | Timestamp | Block Addr. |
> |---|---|---|
>
> Note that this example log view has only three columns even though each log

record contains four categories of information.

(Loaiza, col. 3, lines 52-63).

As illustrated above, Loaiza's solution constructs a table view of the database log files themselves. As the above-quoted language highlights, Loaiza's solution allows one to query the database log files, not the database itself. This is fundamentally different from Applicant's claimed invention which uses the transactional log files to reconstruct and maintain a transactionally consistent view (i.e., "cache view") of all tables in a given database, not a table view of the database log files themselves. Applicant's invention is not concerned with providing the capability to view or query the database log file(s) themselves as a table as with Loaiza's solution, but rather provides a view of the database itself which may be used for performing a read-only transaction (Applicant's specification, paragraph [0047]). These features of a "cache view" that records (and maintains in memory) a transactionally consistent version of a database at a given point in time are also specifically described in Applicant's claims. For example, Applicant's amended claim 1 includes the following limitations:

> 1. In a database system employing a transaction log, an improved method for restoring databases to a consistent version supporting read-only uses, the method comprising:
> providing a shared cache storing database blocks in memory of the database system;
> in response to a read-only transaction of a given database, creating a cache view of the given database using the given database's transaction log by logically undoing transactions which have begun but have yet to commit, said cache view comprising particular database blocks in the shared cache that record a transactionally consistent version of the given database at a given point in time supporting read-only uses;
> temporarily storing any database blocks that overflow said shared cache in a temporary database during use of the cache view by the read-only transaction; and
> performing the read-only transaction using the cache view and returning results of the read-only transaction.

(Applicant's amended claim 1, emphasis added)

Additionally, Applicant's claimed invention also includes use of a temporary database (sometimes referred to in Applicant's specification as a "shadow cache") for temporarily storing database blocks of the cache view that overflow the shared cache.

Applicant's approach for storing blocks that "overflow" the shared cache in a temporary database avoids having to reconstruct the blocks which were modified in order to create the transactionally consistent "cache view" of a given database. Recall that Applicant's approach for creation of a transactionally consistent "cache view" may require logically undoing active transactions at the start of a read-only transaction. The use of a temporary database as a "shadow cache" is advantageous in view of the fact that it would be virtually impossible to reconstruct (at a later point in time) the blocks modified from a logical undo that was performed at the time a read-only transaction begins. Additionally, saving blocks physically undone or redone in the "shadow cache" (temporary database) is a performance optimization to avoid reconstructing these blocks.

The Examiner continues to rely on Hayashi as including the corresponding teachings of a shadow cache, referencing Fig. 6 and col. 4, lines 8-18 of Hayashi. However, the referenced teachings relate to writing the contents of a shared buffer to a database so as to nonvolatilize the contents of the shared buffer. Hayashi describes the optimization of buffer locks used during this nonvolitization process and also describes the making of copies of buffers updated by write transactions during the process of nonvolitization. This is not analogous to the use of a temporary database as a backing store as provided in Applicant's specification and claims. Additionally, unlike Hayashi Applicant's solution does not use a bitmap for overflow blocks. Instead, Applicant uses a temporary database (and more particularly, a high speed look up table that has one row for each overflow block as described, for example, in Applicant's dependent claim 8). Another difference between Applicant's solution and that of Hayashi is that Applicant's solution does not need to be concerned with providing for crash recovery for a bitmap or overflow blocks. Since Applicant's cache views are used for read-only purposes, the cache views are completely discardable in the event of a system crash. Hayashi, in contrast, requires logging and related mechanisms to guard against crash recovery (see e.g., Hayashi, col. 4, lines 19-27).

Another important aspect of Applicant's invention is the use of "logical undos" for reconstructing a transactionally consistent (i.e., proper, as of a given point in time) version of the database. As described in Applicant's previously filed Appeal Brief, in a multi-user database environment, it is often not possible to simply use physical undo and

redo to create a transactionally consistent version of a database at a particular point in time. In a multi-user database environment one may have multiple database users with intermingled write operations occurring against the database, some of which have committed while others are still pending. The use of logical undo enables Applicant's solution to get the database to a transactionally consistent prior state that is suitable for performing long-duration read-only transactions (e.g., DSS, reporting, data mining, or the like).

The Examiner acknowledges that Hayashi includes no teaching of logical undo operations and therefore adds Luo as providing teachings comparable to those of Applicant's claimed invention. However, while Luo describes performing a logical undo, Luo makes no mention of doing so in order to reconstruct a transactionally consistent read-only view of a database. Instead, the referenced teachings of Luo simply provide as follows:

> In accordance with some embodiments, a logical undo is performed. In the logical undo, the database system looks for another tuple in the join view JV that has the attribute values (1, 2). That other tuple is changed to the value (1, 1) for a logical undo of transaction T.sub.1, shown as 156 in FIG. 5A.
>
> (Luo, col. 13, lines 16-21).

There are many reasons for executing logical undo including transaction rollback and crash recovery. Luo's solution is primarily focused on avoiding lock conflicts and deadlocks. Transaction rollback (logical undo) is a common approach to breaking out of deadlock. However, it is not analogous to Applicant's claimed invention which specifically provides for use of logical undo operations to create a transactionally consistent version of a given database as described, for instance, in the following limitations of Applicant's claim 1:

> creating a cache view of the given database using the given database's transaction log by logically undoing transactions which have begun but have yet to commit, said cache view comprising particular database blocks in the shared cache that record a transactionally consistent version of the given database at a given point in time supporting read-only uses;
>
> (Applicant's amended claim 1, emphasis added)

All told Hayashi, Loaiza and Luo, either alone or in combination, do not teach the creation of a transactionally consistent prior version of the database supporting read-only uses. In particular, the combined references do not provide comparable teachings of logically undoing incomplete transactions in order to create a transactionally consistent version ("cache view") of the database in the manner provided in Applicant's specification and claims. Additionally, the references also fail to include teachings analogous to Applicant's claim limitations of creating this "cache view" using the transaction log. Instead, Loaiza describes creating a table view of the transaction log records themselves, rather than using the log records in creating a view of the database itself as provided by Applicant's invention. Therefore, as the prior art references do not teach or suggest all the limitations of Applicant's claims, it is respectfully submitted that Applicant's claimed invention is distinguishable over the prior art of record and the rejection of claims 1-3, 5-8, 10, 12, 13, 15-19, 21-24, 26, 28, and 29 under Section 103 is overcome.

B. Second Rejection under 35 U.S.C. 103(a): Hayashi and IEEE

Claims 11 and 27 are rejected under 35 U.S.C. 103(a) as being unpatentable over Hayashi (above), in view of The Authoritative Dictionary of IEEE Standards Terms, Seventh Edition, IEEE Press, 2000 ("IEEE"). Claims 11 and 27 are dependent upon Applicant's independent claims 1 and 17 and therefore are believed to be allowable for at least the reasons cited above pertaining to the deficiencies of Hayashi (as well as Loaiza and Luo) in respect to Applicant's claimed invention. The IEEE reference does not cure any of these deficiencies of Hayashi as it does not teach anything analogous to Applicant's claim limitations of creating a transactionally consistent "cache view" of a given database at a particular point in time which is maintained in cache and used to support performance of read-only transactions. As the prior art references do not teach or suggest all the limitations of Applicant's claims, it is respectfully submitted that the rejection of Applicant's claims 11 and 27 is overcome.

C. Third Rejection under 35 U.S.C. 103(a): Hayashi and Raz

Claims 14 and 30 are rejected under 35 U.S.C. 103(a) as being unpatentable over Hayashi (above) in view of Raz, U.S. 5,701,480 ("Raz"). Claims 14 and 30 are also dependent upon Applicant's independent claims 1 and 17 and therefore are believed to be allowable for at least the reasons discussed in detail above as to the deficiencies of the Hayashi, Loaiza and Luo references as to Applicant's invention. Additionally, these claims are believed to be distinguishable from the prior art of record for the following additional reasons.

As to claim 14, Applicant's intervening claim 13 includes limitations providing that upon occurrence of write operations, back link log records are added to the database's transaction log that serve to link together blocks of the transaction log that pertain to a particular operation (Applicant's claim 13). Although the Examiner references Hayashi at col. 3, line 66 to col. 4, line 7 for the corresponding teaching, that portion of Hayashi simply describes that when the contents of a buffer are written to disk the buffer holding information about updates made by the last read or write operation are no longer needed. Respectfully Applicant does not believe that these teachings of Hayashi are at all analogous to Applicant's claim 13 (or the similar limitations of Applicant's claim 29)

Additionally, the limitations of claim 14 further provide that if a transaction must be undone, the back link log records (described in claim 13 above) are used to skip irrelevant portions of the transaction log. The Examiner acknowledges that Hayashi does not provide these teachings but states that Raz discloses an "undo" recovery mechanism that provides very fast recovery because only the effects of failed transactions must be undone. The Examiner states that it would have been obvious to one of ordinary skill in the caching art to combine the teachings of the cited references because Hayashi's teachings would have allowed Raz's method and system to provide a back link log record to skip portions of the transaction log so as to enable users to gain the ability to skip irrelevant portions of the transaction log. However, the referenced teachings of Raz do not include the specific teachings of back link log records comparable to those of Applicant's claimed invention. Instead, the referenced portion of Raz provides as follows:

> A considerable amount of processing time, however, is spent flushing updated records to non-volatile state memory and updating the non-volatile snapshot

> memory when each transaction is committed .... For transactions that update the same records for multiple transactions, and transactions that are short and do not update many pages, a considerable fraction of the processing time is wasted by flushing the updated records to state memory at the end of every transaction.
>
> (Raz, col. 62, lines 3-17)

As illustrated above, Raz discusses the problems inherent in writing each and every change made by a transaction to disk. However, Applicant's review of the above portion of Raz (and the balance of the Raz reference) finds no comparable teaching of back linking records in the transaction log relating to particular transactions so that such transactions may be more quickly undone.

Any dependent claims not explicitly discussed are believed to be allowable by virtue of dependency from Applicant's independent claims, as discussed in detail above.

Conclusion

In view of the foregoing remarks and the amendment to the claims, it is believed that all claims are now in condition for allowance. Hence, it is respectfully requested that the application be passed to issue at an early date.

If for any reason the Examiner feels that a telephone conference would in any way expedite prosecution of the subject application, the Examiner is invited to telephone the undersigned at 925 465-0361.

Respectfully submitted,

Date: April 11, 2008

/G. Mack Riddle/

G. Mack Riddle, Reg. No. 55,572
Attorney of Record

925 465-0361
925 465-8143 FAX